Making a Network Orchard by Adding Leaves

van Iersel, Leo; Jones, Mark; Julien, Esther; Murakami, Yukihiro

DOI
10.4230/LIPIcs.WABI.2023.7

Publication date
2023

Document Version
Final published version

Published in
23rd International Workshop on Algorithms in Bioinformatics, WABI 2023

Making a Network Orchard by Adding Leaves

Leo van Iersel
Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands

Mark Jones
Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands

Esther Julien
Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands

Yukihiro Murakami[1]
Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands

Abstract

Phylogenetic networks are used to represent the evolutionary history of species. Recently, the new class of orchard networks was introduced, which were later shown to be interpretable as trees with additional horizontal arcs. This makes the network class ideal for capturing evolutionary histories that involve horizontal gene transfers. Here, we study the minimum number of additional leaves needed to make a network orchard. We demonstrate that computing this proximity measure for a given network is NP-hard and describe a tight upper bound. We also give an equivalent measure based on vertex labellings to construct a mixed integer linear programming formulation. Our experimental results, which include both real-world and synthetic data, illustrate the efficiency of our implementation.

2012 ACM Subject Classification Mathematics of computing → Trees; Mathematics of computing → Graph algorithms; Applied computing → Biological networks; Applied computing → Bioinformatics

Keywords and phrases Phylogenetics, Network, Orchard Networks, Proximity Measures, NP-hardness

Digital Object Identifier 10.4230/LIPIcs.WABI.2023.7

Supplementary Material
Software (Source Code): `https://github.com/estherjulien/OrchardProximity`
archived at `swh:1:dir:d80c03267ac6a1474c350655e337f1ae7de3f945`

Funding *Leo van Iersel*: Research funded in part by Netherlands Organization for Scientific Research (NWO) grants OCENW.KLEIN.125 and OCENW.GROOT.2019.015.
Mark Jones: Research funded by Netherlands Organization for Scientific Research (NWO) grant OCENW.KLEIN.125.
Esther Julien: Research funded by Netherlands Organization for Scientific Research (NWO) grant OCENW.GROOT.2019.015.

1 Introduction

Phylogenetic trees are used to represent the evolutionary history of species. While they are effective for illustrating speciation events through vertical descent, they are insufficient in representing more intricate evolutionary processes. Reticulate (net-like) events such as hybridization and horizontal gene transfer (HGT) can give rise to signals that cannot be represented on a single tree [9, 21]. In light of this, phylogenetic networks have gained increasing attention due to their capability in elucidating reticulate evolutionary processes.

[1] Corresponding author

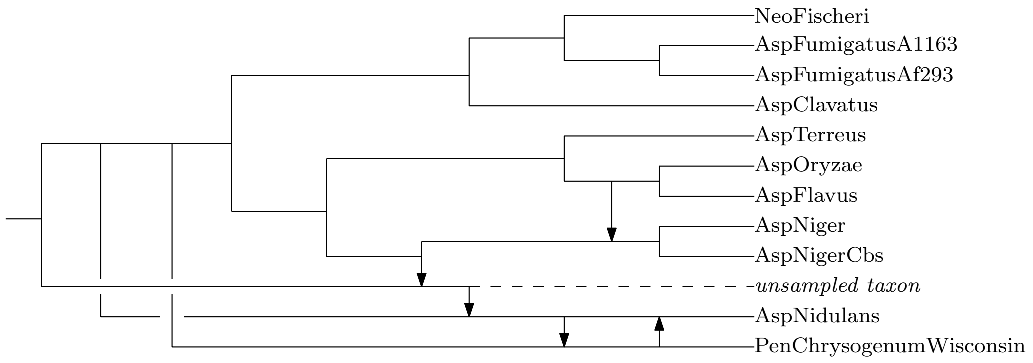


Figure 1 A network on 11 different taxa (excluding *unsampled taxon*) of fungi including 5 reticulations, which is part of a larger network from [18]. The directed arcs in the figure are linking arcs, which represent gene transfer highways. In order to make all linking arcs horizontal, we require an additional leaf (*unsampled taxon*) to represent the evolutionary history. To see that the network needs the leaf *unsampled taxon* in order to have only horizontal linking arcs, we refer the interested reader to Appendix A.

Phylogenetic networks are often categorized into different classes based on their topological features. These are often motivated computationally, but some classes are also defined based on their biological relevance [16]. Classical examples of network classes involve the *tree-child networks* [3] and the *tree-based networks* [8]. Roughly speaking, tree-child networks are those where every vertex has passed on a gene via vertical descent to an extant species, and tree-based networks are those obtainable from a tree by adding so-called *linking arcs* between tree arcs. Recent developments have culminated in the introduction of *orchard networks*, which lie – inclusion-wise – between the two aforementioned network classes [14, 4]. The class has shown to be both algorithmically attractive and biologically relevant; they are defined as networks that can be reduced to a single leaf by a series of so-called *cherry-picking operations*, and they were shown to be networks that can be obtained by adding horizontal arcs to trees (where the tree is drawn with the root at the top and arcs pointing downwards) [19]. Such horizontal arcs can be used to model HGT events, making orchard networks especially apt in representing evolutionary scenarios where every reticulate event is a horizontal transfer. Orchard networks have also been characterized statically based on so-called *cherry covers* [20].

When considering a non-orchard network, a natural question arises: how many additional leaves are required to transform the network into one that is orchard? From a biological standpoint, this question can be interpreted as asking how many extinct species or unsampled taxa need to be introduced into the network to yield a scenario where every reticulation represents an HGT event. Given that HGT is the primary driver of reticulate evolution in bacteria [10], this is an essential inquiry. We provide a network of a few fungi species in Figure 1, which requires one additional leaf to make it orchard. Formally speaking, the problem of computing this leaf addition measure is as follows.

$L_{\mathcal{OR}}$-DISTANCE (DECISION)
Input: A network N on a set of taxa X and a natural number k.
Decide: Can N be made orchard with at most k leaf additions?

In related research, the leaf addition measure has been investigated for other network classes. It has been shown that tracking down the minimum leaf additions to make a network tree-based can be done in polynomial time [7]. In the same paper, it was shown that the leaf addition measure was equivalent to two other proximity measures, namely those based on spanning trees and disjoint path partitions. The same question was posed for the unrooted variant (where the arcs of the network are undirected), for which the problem turned out to be NP-complete [5]. A total of eight proximity measures were introduced in this latter paper, including those based on edge additions and rearrangement moves. Instead of considering leaf additions, some manuscripts have even considered leaf deletions (in general, vertex deletions) as proximity measures for the class of so-called *edge-based networks* [6]. Finally for orchard networks, a recent bachelor's thesis compared how the leaf addition proximity measure differs in general to another proximity measure based on arc deletions [17].

In this paper, we show that the leaf addition proximity measure can be computed in polynomial time for the class of tree-child networks, and we give a more efficient algorithm for computing the measure for tree-based networks. We show that $L_{\mathcal{OR}}$-DISTANCE is NP-complete by a polynomial-time reduction from DEGREE-3 VERTEX COVER. To model the problem as a *mixed integer linear program* (MILP), we consider a reformulation of the leaf addition measure in terms of vertex labellings. Orchard networks are known to be trees with added horizontal arcs; roughly speaking, this means we can label the vertices of an orchard network so that every vertex of indegree-2 has exactly one incoming arc whose end-vertices have the same labels. The reformulated measure, called the *vertical arcs* proximity measure, counts – over all possible vertex labellings (defined formally in Section 6.1) – the minimum number of indegree-2 vertices with only non-horizontal incoming arcs. Our experimental results are promising, as the real world cases are solved in a fraction of a second. Furthermore, the model also scales well to larger synthetic data.

The structure of the paper is as follows. In Section 2, we provide all necessary definitions and characterizations of orchard networks and tree-based networks. In Section 3, we formally introduce the leaf addition measure for the classes of tree-child, orchard, and tree-based networks. In Section 4 we show that $L_{\mathcal{OR}}$-DISTANCE is NP-complete (Theorem 17). In Section 5, we give a sharp upper bound for the leaf addition proximity measure. In Section 6 we give a reformulation of the leaf addition measure to describe the MILP to solve $L_{\mathcal{OR}}$-DISTANCE, and in Section 6.3, experimental results are shown for the MILP, applied to real and simulated networks. In Section 7, we give a brief discussion of our results and discuss potential future research directions. We include proofs for select results in Appendix A.

2 Preliminaries

A *binary directed phylogenetic network* on a non-empty set X is a directed acyclic graph with

- a single *root* of indegree-0 and outdegree-1;
- *tree vertices* of indegree-1 and outdegree-2;
- *reticulations* of indegree-2 and outdegree-1;
- *leaves* of indegree-1 and outdegree-0, that are labelled bijectively by elements of X.

For the sake of brevity, we shall refer to binary directed phylogenetic networks simply as *networks*. Throughout the paper, assume that N is a network on some non-empty set X where $|X| = n$, unless stated otherwise. Networks without reticulations are called *trees*. Tree vertices and reticulations may sometimes collectively be referred to as *internal vertices*.

The arc uv of a network is a *root arc* if u is the root of the network. An arc uv of a network is a *reticulation arc* if v is a reticulation, and a *tree arc* otherwise. We say that a vertex u is a *parent* of another vertex v if uv is an arc of the network; in such instances we call v a *child* of u. Also, we say that u and v are the *tail* and the *head* of the arc uv, respectively. In other words, we may rewrite arcs as $uv = \text{tail}(uv)\,\text{head}(uv)$. The *neighbours* of v refer to the set of vertices that are parents or children of v. We also say that vertices u and v are *siblings* if they share a parent.

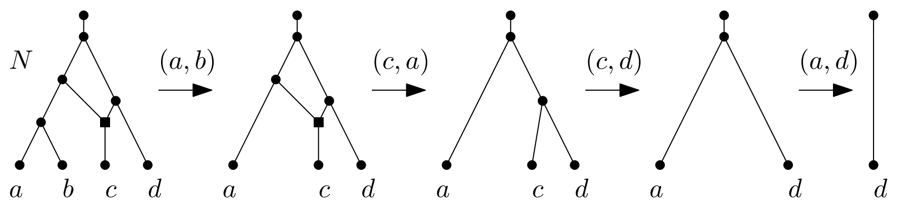


Figure 2 An example of an orchard network N that is reduced by a sequence $(a,b)(c,a)(c,d)(a,d)$. The network N contains a cherry (a,b) and a reticulated cherry (c,d). Subsequent networks are those obtained by a single cherry picking reduction from the previous network. For example, the second network $N(a,b)$ is obtained from N by removing the leaf b and cleaning up. Note that the network is also tree-child.

In what follows, we shall define graph operations based on vertex and arc deletions. To make sure resulting graphs remain networks, we follow-up every graph operation with a *cleaning up* process. Formally, we *clean up* a network by applying the following until none is applicable.

- Suppress an indegree-1 outdegree-1 vertex (e.g., if uv and vw are arcs where v is an indegree-1 outdegree-1 vertex, we suppress v by deleting the vertex v and adding an arc uw).
- Replace parallel arcs by a single arc (e.g., if uv is an arc twice in a network, delete one of the arcs uv).

We observe that deleting a tree arc and cleaning up results in a graph containing two indegree-0 vertices. On the other hand, deleting a reticulation arc and cleaning up results in a network. Therefore, we shall use arc deletions to mean reticulation arc deletions.

2.1 Tree-Child Networks

A network is *tree-child* if every non-leaf vertex has a child that is a tree vertex or a leaf. We call an internal vertex of a network an *omnian* if all of its children are reticulations [15]. It follows from definition that a network is tree-child if and only if it contains no omnians.

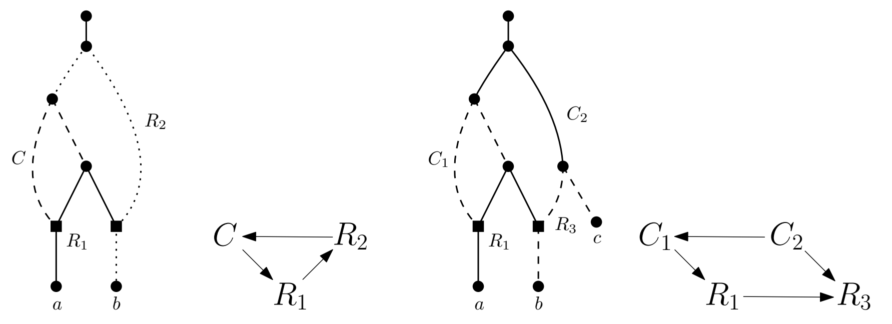
2.2 Orchard Networks

To define orchard networks, we must first define cherries and reticulated cherries, as well as operations to reduce them. See Figure 2 for the illustration of the following definitions. Let N be a network. Two leaves x and y of N form a *cherry* if they are siblings. In such a case, we say that N *contains* a cherry (x,y) or a cherry (y,x). Two leaves x and y of N form a *reticulated cherry* if the parent p_x of x is a reticulation and the parent of y is also a parent of p_x. In such a case, we say that N *contains* a reticulated cherry (x,y). *Reducing the cherry* (x,y) *from* N is the process of deleting the leaf x and cleaning up. *Reducing the reticulated cherry* (x,y) *from* N is the process of deleting the arc from the parent of y to the parent of x and cleaning up. In both cases, we use $N(x,y)$ to denote the resulting network.

A network N is *orchard* if there is a sequence $S=(x_1,y_1)(x_2,y_2)\dots(x_k,y_k)$ such that NS is a network on a single leaf y_k. It has been shown that the order in which (reticulated) cherries are reduced does not matter [4, 14]. Apart from this recursive definition, orchard networks have been characterized based on cherry covers (arc decompositions) [20] and vertex labellings [19]. We include both characterizations here.

Cherry covers (see [20] for more details). A *cherry shape* is a subgraph on three distinct vertices x, y, p with arcs px and py. The *internal vertex* of a cherry shape is p, and the *endpoints* are x and y. A *reticulated cherry shape* is a subgraph on four distinct vertices x, y, p_x, p_y with arcs $p_x x, p_y p_x, p_y y$, such that p_x is a reticulation in the network. The *internal vertices* of a reticulated cherry shape are p_x and p_y, and the endpoints are x and y. The *middle arc* of a reticulated cherry shape is $p_y p_x$. We will often refer to cherry shapes and the reticulated cherry shapes by their arcs (e.g., we would denote the above cherry shape $\{p_x x, p_y y\}$ and the reticulated cherry shape $\{p_x x, p_y p_x, p_y y\}$). We say that an arc uv is covered by a cherry or reticulated cherry shape B if $uv \in B$. A *cherry cover* of a binary network is a set P of cherry shapes and reticulated cherry shapes, such that each arc except for the root arc is covered exactly once by P. In general, a network can have more than one cherry cover.

We define the *cherry cover auxiliary graph* $G = (V, A)$ of a cherry cover as follows. For all shapes $B \in P$, we have $v_B \in V$. A shape $B \in P$ is *directly above* another shape $C \in P$ if B and C contain a same vertex v, such that v is an endpoint of B and an internal vertex of C. Then, $v_B v_C \in A$ (adapted from [20, Definition 2.13]). We say that a cherry cover is *cyclic* if its auxiliary graph has a cycle. We call it *acyclic* otherwise. See Figure 3 for an illustration of a cyclic and acyclic cherry cover.



(a) Network N_1. **(b)** Cherry cover aux. graph of N_1. **(c)** Network N_2. **(d)** Cherry cover aux. graph of N_2.

Figure 3 A cherry cover example. (a) A network N_1 on $\{a, b\}$ with a cherry cover $\{C, R_1, R_2\}$. (b) The (cyclic) auxiliary graph of N_1 based on the cherry cover of (a). (c) The network N_2 obtained from N_1 by adding a leaf c, with a cherry cover $\{C_1, C_2, R_1, R_3\}$ (d) The (acyclic) auxiliary graph of N_2 based on the cherry cover of (d).

▶ **Theorem 1** (Theorem 4.3 of [20]). *A network N is orchard if and only it has an acyclic cherry cover.*

Non-Temporal Labellings. Let N be a network with vertex set $V(N)$. A *non-temporal labelling*[2] of N is a labelling $t : V(N) \to \mathbb{R}$ such that

- for all arcs uv, $t(u) \leq t(v)$ and equality is allowed only if v is a reticulation;
- for each internal vertex u, there is a child v of u such that $t(u) < t(v)$;
- for each reticulation r with parents u and v, at most one of $t(u) = t(r)$ or $t(v) = t(r)$ holds.

[2] This is named in contrast to *temporal representations* of [1]. There, it was required for the endpoints of every reticulation arc to have the same label.

Observe that every network (orchard or not) admits a non-temporal labelling by labelling each vertex by its longest distance from the root (assuming each arc is of weight 1).

Under non-temporal labellings, we call an arc *horizontal* if its endpoints have the same label; we call an arc *vertical* otherwise. By definition, only reticulation arcs can be horizontal. We say that a non-temporal labelling is an *HGT-consistent labelling* if every reticulation is incident to exactly one incoming horizontal arc. We recall the following key result.

▶ **Theorem 2** (Theorem 1 of [19]). *A network is orchard if and only if it admits an HGT-consistent labelling.*

2.3 Tree-Based Networks

A network N is *tree-based* with *base tree* T if it can be obtained from T in the following steps.

1. Replace some arcs of T by paths, whose internal vertices we call *attachment points*; each attachment point is of indegree-1 and outdegree-1.
2. Place arcs between attachment points, called *linking arcs*, so that the graph contains no vertices of total degree greater than 3, and so that it remains acyclic.
3. Clean up.

The relation between the classes of tree-child, orchard, and tree-based networks can be stated as follows.

▶ **Lemma 3** ([14] and Corollary 1 of [19]). *If a network is tree-child, then it is orchard. If a network is orchard, then it is tree-based.*

We include here a static characterization of tree-based networks based on an arc partition, called *maximum zig-zag trails* [11, 23]. Let N be a network. A *zig-zag trail* of length k is a sequence $(a_1, a_2, \dots, a_k)$ of arcs where $k \geq 1$, where either $\text{tail}(a_i) = \text{tail}(a_{i+1})$ or $\text{head}(a_i) = \text{head}(a_{i+1})$ holds for $i \in [k-1] = \{1, 2, \dots, k-1\}$. We call a zig-zag trail Z *maximal* if there is no zig-zag trail that contains Z as a subsequence. Depending on the nature of $\text{tail}(a_1)$ and $\text{tail}(a_k)$, we have four possible maximal zig-zag trails.

- *Crowns*: $k \geq 4$ is even and $\text{tail}(a_1) = \text{tail}(a_k)$ or $\text{head}(a_1) = \text{head}(a_k)$.
- *M-fences*: $k \geq 2$ is even, it is not a crown, and $\text{tail}(a_i)$ is a tree vertex for every $i \in [k]$.
- *N-fences*: $k \geq 1$ is odd and $\text{tail}(a_1)$ or $\text{tail}(a_k)$, but not both, is a reticulation. By reordering the arcs, assume henceforth that $\text{tail}(a_1)$ is a reticulation and $\text{tail}(a_k)$ a tree vertex.
- *W-fences*: $k \geq 2$ is even and both $\text{tail}(a_1)$ and $\text{tail}(a_k)$ are reticulations.

We call a set S of maximal zig-zag trails a *zig-zag decomposition* of N if the elements of S partition all arcs, except for the root arc, of N.

▶ **Lemma 4** (adapted from Corollary 4.6 of [11]). *Let N be a network. Then N is tree-based if and only if it has no W-fences.*

▶ **Theorem 5** (adapted from Theorem 4.2 of [11]). *Any network N has a unique zig-zag decomposition.*

▶ **Theorem 6** (adapted from Theorem 3.3 of [20]). *Let N be a network. Then N is tree-based if and only if it has a cherry cover.*

3 Leaf Addition Proximity Measure

Let N be a network on X. *Adding a leaf* $x \notin X$ *to an arc* e *of* N is the process of adding a labelled vertex x, subdividing the arc e by a vertex w (if $e = uv$ then we delete the arc uv, add the vertex w, and add arcs uw and wv), and adding an arc wx. We denote the resulting network by $N + (e, x)$. When the arc e in the above is irrelevant or clear, we simply call this process *adding a leaf* x *to* N, and denote the resulting network by $N + x$.

In this section, $\mathcal{C}$ will be used to denote a network class. In particular, we shall use $\mathcal{TC}, \mathcal{OR}$, and $\mathcal{TB}$ to denote the classes of tree-child networks, orchard networks, and tree-based networks, respectively. Let $L_{\mathcal{C}}(N)$ denote the minimum number of leaf additions required to make the network N a member of $\mathcal{C}$. We first show that computing $L_{\mathcal{TC}}(N)$ and $L_{\mathcal{TB}}(N)$ can be done in polynomial time.

▶ **Lemma 7.** *Let N be a network. Then $L_{\mathcal{TC}}(N)$ is equal to the number of omnians. Moreover, N can be made tree-child by adding a leaf to exactly one outgoing arc of each omnian.*

▶ **Lemma 8.** *Let N be a network. Then $L_{\mathcal{TC}}(N)$ can be computed in $O(|N|)$ time.*

It has been shown already that $L_{\mathcal{TB}}(N)$ can be computed in $O(|N|^{3/2})$ time where $|N|$ is the number of vertices in N [7]. We show that this can in fact be computed in $O(|N|)$ time.

▶ **Lemma 9.** *Let N be a network. Then $L_{\mathcal{TB}}(N)$ is equal to the number of W-fences. Moreover, N is tree-based by adding a leaf to any arc in each W-fence in N.*

▶ **Lemma 10.** *Let N be a network. Then $L_{\mathcal{TB}}(N)$ can be computed in $O(|N|)$ time.*

Interestingly, computing $L_{\mathcal{OR}}(N)$ proves to be a difficult problem, although the leaf addition proximity measure is easy to compute for its neighbouring network classes. We prove the following in Section 4.

▶ **Theorem 17.** Let N be a network. Computing $L_{\mathcal{OR}}(N)$ is NP-hard.

We also include the following theorem which states that when considering leaf addition proximity measures for orchard networks, it suffices to consider leaf additions to reticulation arcs. We shall henceforth assume that all leaf additions are on reticulation arcs.

▶ **Theorem 11** (Theorem 4.1 of [17]). *A network N is orchard if and only if the network obtained by adding a leaf to a tree arc of N is orchard.*

The rest of the paper will now focus on the problem of computing $L_{\mathcal{OR}}(N)$.

4 Hardness Proof

In this section, we show that computing $L_{\mathcal{OR}}(N)$ is NP-hard by reducing from degree-3 vertex cover.

DEGREE-3 VERTEX COVER (DECISION)
Input: A 3-regular graph $G = (V, E)$ and a natural number k.
Decide: Does G have a vertex cover with at most k vertices?

$L_{\mathcal{OR}}$-DISTANCE (DECISION)
Input: A network N on a set of taxa X and a natural number k.
Decide: Can N be made orchard with at most k leaf additions?

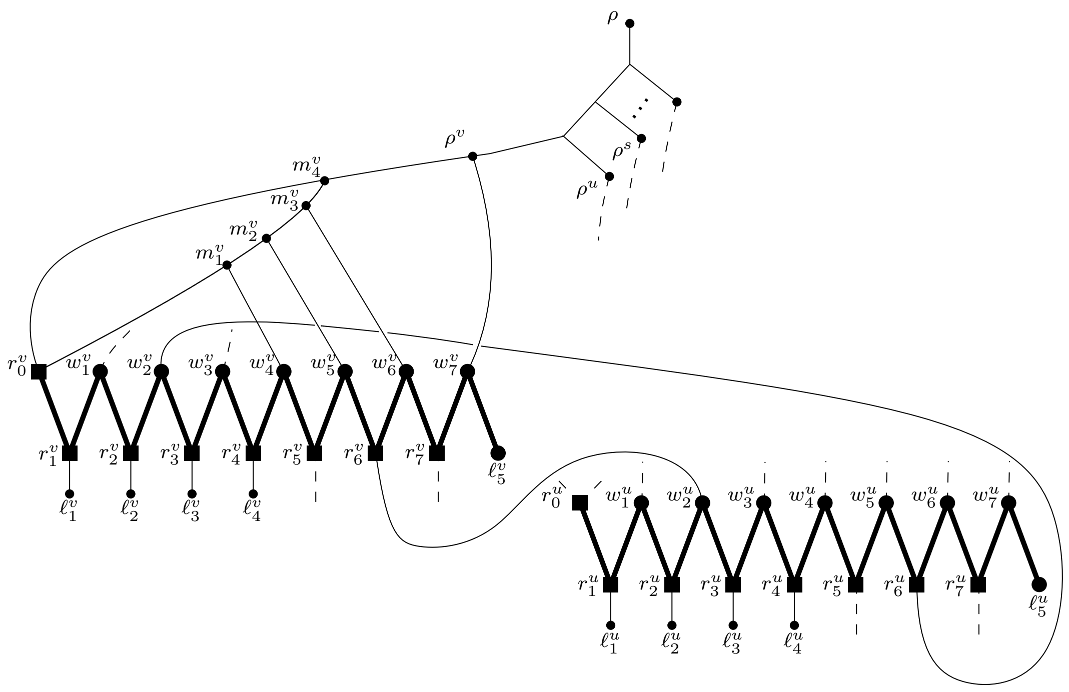

ρ
ρ^s
ρ^u
ρ^v
m_4^v
m_3^v
m_2^v
m_1^v
r_0^v w_1^v w_2^v w_3^v w_4^v w_5^v w_6^v w_7^v
r_1^v r_2^v r_3^v r_4^v r_5^v r_6^v r_7^v ℓ_5^v
ℓ_1^v ℓ_2^v ℓ_3^v ℓ_4^v
r_0^u w_1^u w_2^u w_3^u w_4^u w_5^u w_6^u w_7^u
r_1^u r_2^u r_3^u r_4^u r_5^u r_6^u r_7^u ℓ_5^u
ℓ_1^u ℓ_2^u ℓ_3^u ℓ_4^u

Figure 4 Sketch of the network N_G for the case when G contains an edge uv.

We now describe the reduction from Degree-3 Vertex Cover to $L_{\mathcal{OR}}$-Distance. For a graph G, let $V(G)$ and $E(G)$ be its vertex and edge sets, respectively. Given an instance (G, k) of Degree-3 Vertex Cover, construct an instance (N_G, k) of $L_{\mathcal{OR}}$-Distance as follows (see Figure 4):

1. For each vertex v in $V(G)$, construct a gadget Gad(v) as described below. In what follows, vertices of the form ℓ_i^v are leaves, vertices r_i^v are reticulations, and vertices w_i^v, m_i^v and ρ^v are tree vertices.
   The key structure in Gad(v) is an N-fence with 15 arcs, starting with the arc $\boldsymbol{r_0^v r_1^v}$, then followed by arcs of the form $\boldsymbol{w_i^v r_i^v, w_i^v r_{i+1}^v}$ for each $i \in [6]$, and finally the arcs $\boldsymbol{w_7^v r_7^v, w_7^v \ell_5^v}$. This set of arcs, in bold type, is called the *principal part* of Gad(v). In addition,the reticulations $r_1^v, r_2^v, r_3^v, r_4^v$ have leaf children $\ell_1^v, \ell_2^v, \ell_3^v, \ell_4^v$ respectively.
   Above the principal part of Gad(v), add a set of tree vertices $m_1^v, m_2^v, m_3^v, m_4^v, \rho^v$ with the following children: m_1^v has children r_0^v and w_4^v, m_2^v has children m_1^v and w_5^v, m_3^v has children m_2^v and w_6^v, m_4^v has children m_3^v and r_0^v, and ρ^v has children m_4^v and w_7^v (see Figure 4).
   This completes the construction of Gad(v). Note that so far, the vertices w_1^v, w_2^v, w_3^v have no incoming arcs, and r_5^v, r_6^v, r_7^v have no outgoing arcs. Such arcs will be added later to connect different gadgets together.
2. Connect the vertices ρ^v from each Gad(v) as follows: take some ordering of the vertices $\{v_1, \ldots, v_g\}$ of G. Add a vertex ρ and vertices s_i for $i \in [g-1]$. Add arcs ρs_1 and also arcs from the set $\{s_i s_{i+1} : i \in [g-2]\}$, as well as arcs from the set $\{s_i \rho^{v_i} : i \in [g-1]\}$, and finally an arc $s_{g-1}\rho^{v_g}$.
3. Next add arcs between the gadgets corresponding to adjacent vertices in G, in the following way: for every pair of adjacent vertices u, v in G, add an arc connecting one of the vertices r_5^u, r_6^u, r_7^u in Gad(u) to one of the vertices w_1^v, w_2^v, w_3^v in Gad(v) (and, symmetrically, an arc connecting one of r_5^v, r_6^v, r_7^v to one of w_1^u, w_2^u, w_3^u). The exact choice

of vertices connected by an arc does not matter, except that we should ensure each vertex is used by such an arc exactly once. Formally: for each vertex v in G with neighbours a, b, c, fix two (arbitrary) mappings $\pi_v : \{a, b, c\} \to \{1, 2, 3\}$ and $\tau_v : \{a, b, c\} \to \{5, 6, 7\}$. Then for each pair of adjacent vertices u, v in G, add an arc from $r^u_{\tau_u(v)}$ to $w^v_{\pi_v(u)}$ (and, symmetrically, add an arc from $r^v_{\tau_v(u)}$ to $w^u_{\pi_u(v)}$).

4. Finally, for each vertex v in G, label the vertices $\{\ell^v_i : i \in [5]\}$ in $\mathrm{Gad}(v)$ by ℓ^v_i.

Call the resulting graph N_G; it is easy to see that N_G is directed and acyclic with a single root ρ. Therefore it is a network on the leaf-set $\{\ell^v_i : i \in [5],\ v \in V(G)\}$. As the arcs of N_G are decomposed into M-fences and N-fences, we have the following observation.

▶ **Observation 12.** *Let G be a 3-regular graph and let N_G be the network obtained by the reduction. Then N_G is tree-based.*

By Observation 12 and Theorem 6, we use freely from now on that N_G has a cherry cover. Before proving the main result, we require some notation and helper lemmas. Let N be a network and let $\hat{N}_i$ be an N-fence of N. In what follows, we shall write $\hat{N}_i := (a^i_1, a^i_2, \ldots, a^i_{k_i})$, and we will let c^i_{2j-1} denote the child of $\mathrm{head}(a^i_{2j-1})$ for $j \in \left[\frac{k_i-1}{2}\right]$. The first lemma states that although a tree-based network may have non-unique cherry covers, the reticulated cherry shapes that cover arcs of N-fences are fixed.

▶ **Lemma 13.** *Let N be a tree-based network, and let $\hat{N}_1, \hat{N}_2, \ldots, \hat{N}_n$ denote the N-fences of N of length at least 3. Then every cherry cover of N contains the reticulated cherry shapes $\{(\mathrm{head}(a^i_{2j-1})c^i_{2j-1}), a^i_{2j}, a^i_{2j+1}\}$ for $i \in [n]$ and $j \in \left[\frac{k_i-1}{2}\right]$.*

Note that the principal part of a gadget $\mathrm{Gad}(v)$ for every $v \in V(G)$ is an N-fence. Let us denote the principal part of a gadget $\mathrm{Gad}(v)$ by $(a^v_1, a^v_2, \ldots, a^v_{15})$ for all $v \in V(G)$. By Lemma 13, a^v_i for $i = 2, \ldots, 15$ and $v \in E(G)$ are covered in the same manner across all possible cherry covers of N_G. Let us denote the reticulated cherry shape that contains a^v_i and a^v_{i+1} by $R^v_{i/2}$ for even $i \in [15]$. Figures 5a and 5b show an example of the part of cherry cover auxiliary graph containing R^v_i and R^u_i for $i \in [7]$, for some edge uv in G. Note that the cherry shapes form a cycle. The next lemma implies that in fact, such a cycle exists for any edge uv in G.

▶ **Lemma 14.** *Let N be a tree-based network and suppose that for two N-fences $\hat{N}_u := (a^u_1, a^u_2, \ldots, a^u_{k_u})$ and $\hat{N}_v := (a^v_1, a^v_2, \ldots, a^v_{k_v})$ of length at least 3, there exist directed paths in N from $\mathrm{head}(a^u_h)$ to $\mathrm{tail}(a^v_i)$ and from $\mathrm{head}(a^v_j)$ to $\mathrm{tail}(a^u_k)$, for even h, i, j, k with $k < h$ and $i < j$. Then every cherry cover auxiliary graph of N contains a cycle.*

In order to remove all possible cycles from a possible cherry cover, it is therefore necessary to disrupt the principal part of either $\mathrm{Gad}(u)$ or $\mathrm{Gad}(v)$, for any edge uv in G.

▶ **Lemma 15.** *Let G be a 3-regular graph and let N_G be the network obtained by the reduction above. Suppose that A is a set of arcs of N_G, for which adding leaves to every arc in A results in an orchard network. For every edge $uv \in E(G)$, there exists an arc $a \in A$ that is an arc of the principal part of $\mathrm{Gad}(u)$ or $\mathrm{Gad}(v)$.*

To complete the proof of the validity of the reduction, we show that in order to make N_G orchard by leaf additions, it is sufficient (and necessary) to add a leaf z^v to an appropriate arc of $\mathrm{Gad}(v)$ for every v in a vertex cover V_{sol} of G (see Figure 5c). The key idea is that this splits the principal part of $\mathrm{Gad}(v)$ from an N-fence into an N-fence and an M-fence, and this allows us to avoid the cycle in the cherry cover auxiliary graph (see Figure 5d).

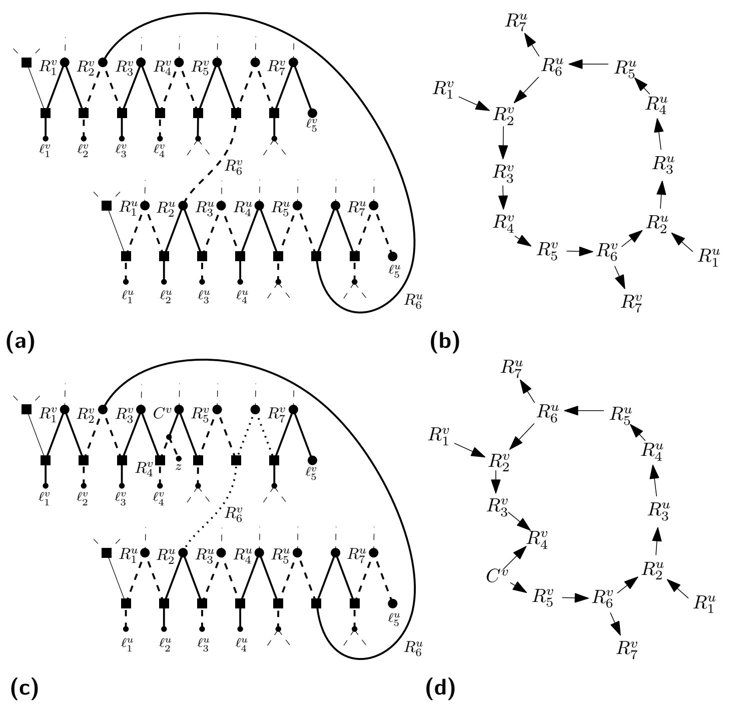


Figure 5 Cherry cover of Gad(v) and Gad(u). In (a), the unique cherry cover of the principal part of Gad(v) and Gad(u) is displayed, in (b), the cherry cover auxiliary graph of (a) is given. In (c), the leaf $z \notin X$ is added to the principal part of Gad(v), and one possible cherry cover of the same part of the network is given. And in (d), the cherry cover auxiliary graph of (c) is given.

▶ **Lemma 16.** *Let G be a 3-regular graph and let N_G be the network obtained by the reduction described above. Then G has a minimum vertex cover of size k if and only if $L_{\mathcal{OR}}(N_G) = k$.*

▶ **Theorem 17.** *Let N be a network. The decision problem $L_{\mathcal{OR}}$-Distance is NP-complete. Computing $L_{\mathcal{OR}}(N)$ is NP-hard.*

Proof. Suppose we are given a set of arcs A_{sol} of N_G of size at most k. Upon adding leaves to every arc in A_{sol}, we may check that the resulting network is orchard in polynomial time (see Section 6 of [14]). This implies that $L_{\mathcal{OR}}$-Distance is in NP. The reduction from Degree-3 Vertex Cover to $L_{\mathcal{OR}}$-Distance outlined at the start of the section takes polynomial time, since we add a constant number of vertices and arcs for every vertex in the Degree-3 Vertex Cover instance. The NP-completeness of $L_{\mathcal{OR}}$-Distance follows from the equivalence of the two problems shown by Lemma 16. The optimization problem of $L_{\mathcal{OR}}$-Distance, i.e., the one of computing $L_{\mathcal{OR}}(N)$ is therefore NP-hard. ◀

5 Upper Bound

In the previous section we showed that computing $L_{\mathcal{OR}}(N)$ is NP-hard. Here, we provide a sharp upper bound for $L_{\mathcal{OR}}(N)$. We call a reticulation *highest* if it has no reticulation ancestors.

▶ **Lemma 18.** *Let N be a network. Suppose there is a highest reticulation r such that all other reticulations have a leaf sibling. Then N is orchard.*

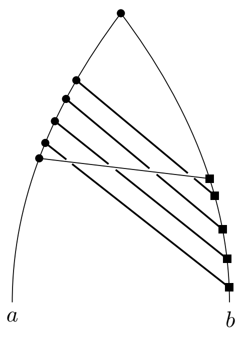


Figure 6 A network N on two leaves $\{a, b\}$ with $r(N) = 5$ reticulations. Observe that $L_{\mathcal{OR}}(N) = r(N) - 1 = 4$, since the highest reticulation cannot be reduced by cherry picking unless the reticulations below it are first reduced. For each non-highest reticulation, we must add a leaf to one of its incoming arcs to reduce it, which leads to $L_{\mathcal{OR}}(N) = 4$. Note that this construction can be extended for any k reticulations.

Proof. We prove the lemma by induction on the number of reticulations k. For the base case, observe that a network with one reticulation is tree-child since it has no omnians. A tree-child network is orchard [14], and so this network must be orchard.

Suppose now that we have proven the lemma for all networks with fewer than k reticulations, where $k > 1$. Let N be a network with reticulation set R where $|R| = k$, and suppose there exists a highest reticulation r in N such that all other reticulations have a leaf sibling. Let r denote the highest reticulation as specified in the statement of the lemma. Choose a lowest reticulation $r' \in R \setminus \{r\}$. By assumption, r' has a leaf sibling c. Every vertex below r' must be tree vertices and leaves. Reduce cherries until the child x of r' is a leaf. Then (x, c) is a reticulated cherry; the network N' obtained by reducing this reticulated cherry has $k - 1$ reticulations and has a highest reticulation r such that all other reticulations have a leaf sibling. By induction hypothesis, N' must be orchard. Since a sequence of cherry reductions can be applied to N to obtain N', the network N must also be orchard. ◀

▶ **Theorem 19.** *Let N be a network, and let $r(N)$ denote the number of reticulations. Then $L_{\mathcal{OR}}(N) = 0$ if N is a tree, and otherwise, $L_{\mathcal{OR}}(N) \leq r(N) - 1$, where the bound is sharp.*

Proof. If N is a tree, then it is orchard, and so $L_{\mathcal{OR}}(N) = 0$. So suppose $r(N) > 0$. Let r be a highest reticulation of N, and for every other reticulation, arbitrarily choose one incoming reticulation arc. Add a leaf to each of these reticulation arcs. By Lemma 18, the resulting network must be orchard. We have added a leaf for all but one reticulation in N. It follows that $L_{\mathcal{OR}}(N) \leq r(N) - 1$. The network in Figure 6 shows that this upper bound is sharp. ◀

6 MILP Formulation

To model the problem of computing the leaf addition proximity measure as a MILP, we reformulate the measure in terms of non-temporal labellings.

6.1 Vertical Arcs into Reticulations

By Theorem 2, every orchard network can be viewed as a network with a base tree where each of the linking arcs are horizontal. Recall that in terms of non-temporal labellings, this means that there exists a labelling wherein every reticulation has exactly one incoming reticulation arc that is horizontal. Following this definition, we introduce a second orchard proximity measure. Given a non-temporal labelling for a network N, let us use *inrets* to refer to reticulations of N with only vertical incoming arcs. Let $V_{\mathcal{OR}}(N)$ denote the minimum number of inrets over all possible non-temporal labellings.

▶ **Observation 20.** *Let N be a network. A network admits an HGT-consistent labelling if and only if $V_{\mathcal{OR}}(N) = 0$. In other words, a network is orchard if and only if $V_{\mathcal{OR}}(N) = 0$.*

In particular, we show a stronger result that equates the two proximity measures.

▶ **Lemma 21.** *Let N be a network. Then $L_{\mathcal{OR}}(N) = V_{\mathcal{OR}}(N)$.*

Proof. Suppose first that we have a network N with some non-temporal labelling $t : V(N) \to \mathbb{R}$ which gives rise to h inrets. For every inret r with parents u and v, we add a leaf x to the arc ur (this addition is done without loss of generality; the argument also follows by adding the leaf to vr). Since r is an inret, we must have $t(u) < t(r)$ and $t(v) < t(r)$. Letting p_x denote the parent of x, we label $t(p_x) := t(r)$ and $t(x) := t(p_x) + 1$. This ensures that the extension of the map t that includes x and p_x is a non-temporal labelling for $N + x$. Observe that r is no longer an inret in $N + x$, since the arc $p_x r$ is horizontal. Therefore, a leaf addition to an incoming arc of an inret can reduce the number of inrets by exactly one. By repeating this procedure for every inret, it follows that $L_{\mathcal{OR}}(N) \le V_{\mathcal{OR}}(N)$.

To show the other direction, suppose we can add ℓ leaves to N to make it orchard. By Theorem 11, we may assume all such leaves are added to reticulation arcs in the set $\{e_1, \ldots, e_\ell\}$. The resulting network N' has an HGT-consistent labelling $t : V(N') \to \mathbb{R}$ by Theorem 2.

We claim that the labelling $t|_{V(N)}$ restricted to N is a non-temporal labelling, and that under $t|_{V(N)}$, the number of inrets is at most ℓ. Suppose that a leaf x_i was added to the reticulation arc $e_i = u_i r_i$. Let p_i denote the parent of x_i in the network N'. By definition of HGT-consistent labellings, we must have that $t(u_i) < t(r_i)$, since $u_i p_i r_i$ is a path in N'. Therefore, restricting the labelling to the network obtained from N' by removing the leaf x_i is non-temporal. Furthermore, if v_i is the parent of r_i that is not u_i, we have that one of $v_i r_i$ or $p_i r_i$ must be horizontal in N'. If $v_i r_i$ was horizontal, then r_i still has a horizontal incoming arc upon removing x_i, and the number of inrets does not change. On the other hand, if $p_i r_i$ was horizontal, then $v_i r_i$ must have been a vertical arc. Upon deleting x_i, the reticulation r_i becomes an inret as its other incoming arc $u_i r_i$ is also vertical. Since leaf deletions are local operations, deleting a leaf increases the number of inrets by at most one. By repeating this for each reticulation arc e_i for $i \in [\ell]$, it follows that N contains at most ℓ inrets, and therefore $V_{\mathcal{OR}}(N) \le L_{\mathcal{OR}}(N)$. ◀

6.2 MILP Formulation

By Lemma 21 we have that $L_{\mathcal{OR}}(N) = V_{\mathcal{OR}}(N)$. In this section, we introduce a MILP formulation to obtain $V_{\mathcal{OR}}(N)$, and therefore also $L_{\mathcal{OR}}(N)$. This is done by searching for a non-temporal labelling of networks in which the number of vertical arcs is minimized.

Let N be a given network with vertex set V and arc set A. Let R denote the set of reticulations of N. We define the decision variable l_v to be the non-temporal label of the vertex $v \in V$. A tree arc and a vertical linking arc uv have the property that $l_u < l_v$. We define x_a to be one if arc $a \in A$ is vertical and zero otherwise. We define h_v to be one if $v \in R$ is a reticulation with only incoming vertical arcs and zero otherwise. Let $v \in V$ be a vertex of N. In what follows, let $P_v \subset V$ be the set of parent nodes of v, $C_v \subset V$ the set of children nodes of v, and X the set of leaves. Let ρ be the root of N. Then, the MILP formulation is as follows:

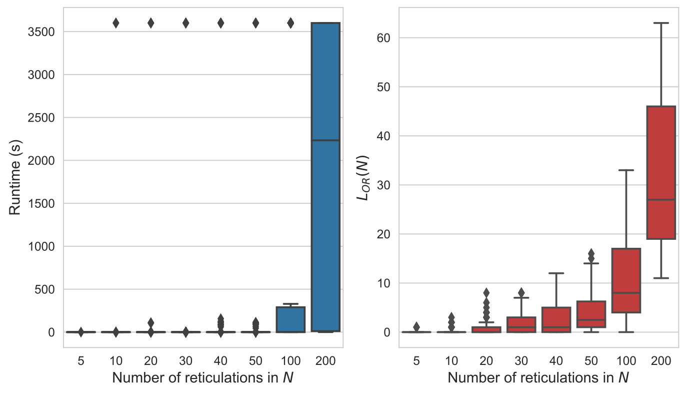
$$
\begin{array}{lll r}
\min\limits_{x,h,l} & \sum\limits_{v \in R} h_v & & \\
\text{s.t.} & \sum\limits_{u \in P_v} x_{uv} - 1 \leq h_v & \forall v \in R & (1) \\
& \sum\limits_{v \in C_u} x_{uv} \geq 1 & \forall u \in V \setminus X & (2) \\
& \sum\limits_{u \in P_v} x_{uv} \geq 1 & \forall v \in V \setminus \{\rho\} & (3) \\
& l_u \leq l_v & \forall uv \in A & (4) \\
& l_u \leq l_v - 1 & \forall v \in V \setminus R, \forall u \in P_v & (5) \\
& l_u \leq l_v - 1 + |V|(1 - x_{uv}) & \forall v \in R, \forall u \in P_v & (6) \\
& l_u \geq l_v - |V| x_{uv} & \forall v \in R, \forall u \in P_v & (7) \\
& x_a \in \{0,1\} & \forall a \in A & \\
& h_v \in \{0,1\} & \forall v \in R & \\
& l_v \in \mathbb{R}_+ & \forall v \in V &
\end{array}
$$

With constraint (1), h_v becomes one if all incoming arcs of reticulation v are vertical. With (2) we have that all vertices must have at least one outgoing vertical arc. Then, (3) guarantees that each reticulation has at least one incoming vertical arc. Constraint (4) creates the non-temporal labelling in the network, where with (5) the label of u is strictly smaller than that of v if v is not a reticulation. Then, (6) sets x_{uv} to one if uv is vertical, for all reticulation vertices v. Finally, with (7) the labels of u and v become equal if x_{uv} is zero.

6.3 Experimental Results

In this section, we apply the MILP described in the previous section to a set of real binary networks and to simulated networks, in order to assess the practical running time. The code for these experiments is written in Python and is available at `https://github.com/estherjulien/OrchardProximity`. All experiments ran on an Intel Core i7 CPU @ 1.8 GHz with 16 GB RAM. For solving the MILP problems, we use the open-source solver SCIP [2].

The real data set consists of different binary networks found in a number of papers, collected on `http://phylnet.univ-mlv.fr/recophync/networkDraw.php`. These networks have a leaf set of size up to 39 and a number of reticulations up to 9, with one outlier that has 32 reticulations. All the binary instances completed within one second (at most 0.072 seconds). Based on the results, we observe that only two out of the 22 binary networks have a value of $L_{\mathcal{OR}}(N) > 0$, thus, that only two are non-orchard. The most interesting of these is the network from [18] since its reticulations represent HGT highways. Even though each highway represents many gene transfers, it is still natural to expect these highways to be horizontal. However, our experimental results show that this network is not orchard (see Appendix A for mathematical arguments) and that its $L_{\mathcal{OR}}$ distance is 1. The most interesting part of this network is redrawn in Figure 1, where we also indicate a way to draw



R	Compl. (%)
5	100.0
10	99.2
20	98.8
30	98.0
40	97.2
50	97.6
100	75.6
200	61.6

(a) **(b)**

Figure 7 Results of simulated networks. (a) Box plots for the runtime results (blue plot) and the $L_{\mathcal{OR}}(N)$ solutions (red plot) per number of reticulations of simulated networks N. The box plots are drawn with respect to the median of the runtime and the leaf addition score. (b) A table with the percentage of instances that were solved within the one-hour time limit. In the plots of (a), we also included instances that did not complete within the one-hour time limit. For these instances, we set their runtime to one hour.

it as a tree with horizontal linking arcs, after adding a single leaf. This added leaf represents a hypothesised missing taxon. In general, the $L_{\mathcal{OR}}(N)$ value gives a lower bound on the number of missing taxa that needs to be added to a network to make it HGT-consistent.

The simulated data is generated using the birth-hybridization network generator of [22], which can generate all binary network topologies [13]. Hence, even though it uses a model with hybridization to construct networks, it can also generate, for example, networks where reticulations represent HGT. This generator has two user-defined parameters: λ, which regulates the speciation rate, and ν, which regulates the hybridization rate. Following [13] we set $\lambda = 1$ and we sampled $\nu \in [0.0001, 0.4]$ uniformly at random. We generated an instance group of size 50 for each pair of values (L, R), with the number of leaves $L \in \{20, 50, 100, 150, 200\}$ and the number of reticulations $R \in \{5, 10, 20, 30, 40, 50, 100, 200\}$. In our implementation, we only defined variables x_a for incoming reticulation arcs. Therefore, the number of binary variables only depends on the number of reticulations in the network. In Figure 7a, the runtime and $L_{\mathcal{OR}}(N)$ value results for the simulated instances are shown against the reticulation number of the networks. The time limit was set to one hour. We can observe from these results that for networks with up to 50 reticulations, almost all instances are solved to optimality within a second. Then for $R = 100, 200$ the runtime increases, mainly because only 75.6% and 61.6% of the instances could be solved within the time limit, respectively (see Figure 7b). The completed instances are often still solved within reasonable time.

7 Discussion

In this paper we investigated the minimum number of leaf additions needed to make a network orchard, as a way to measure the extent to which a network deviates from being orchard. We showed that computing this measure is NP-hard (Theorem 17), and give a sharp upper bound by the number of reticulations minus one (Theorem 19). The measure was reformulated to one in terms of minimizing the number of inrets over all possible non-temporal labellings. In Section 6 we use this reformulation to model the problem of computing the leaf addition measure as a MILP. Experimental results show that real-world data instances were solved within a second and the formulation worked well also over synthetic instances, being able to solve almost all instances up to 50 reticulations and 200 leaves within one second. For bigger instances the runtime however increased.

In this paper we have simulated networks using the network generator of [22] in order to analyse the running time of our MILP. Alternatively, one could simulate networks by generating orchard networks and deleting leaves from them. Since the leaf addition score is finite for any network by Theorem 19, it is possible to obtain any network by using this method. The leaf addition score gives a lower bound on the number of leaves that must be added to make the network orchard. The actual number of missing leaves could be larger, but this value cannot be estimated from the leaf-deleted network.

Of interest is how these results can potentially be used in practice. As mentioned in Section 1, one can consider a scenario in which it is suspected *a priori* that species under consideration evolve under a network in which all reticulate events are horizontal. An example of such scenarios can be seen for horizontal gene transfers, for instance when one considers the evolutionary history of species in bacteria [10] and fungi [18]. If a produced network does not admit an HGT-consistent labelling, there can in general be many reasons. For one, the output network may not be accurate. It is also possible that certain species have gone extinct, or that undersampling is present in the taxon set. In these latter two potential causes, our method gives a way of quantifying the minimum number of taxa that may have gone extinct / been undersampled. Moreover, it can be used to find all optimal corresponding orchard networks with added leaves. This could, for example, be used to try to identify the missing taxa.

Our NP-hardness result is interesting when comparing it to the computational complexity of the corresponding problem for different network classes. The problem of finding the minimum number of leaves to add to make a network tree-based can be solved in polynomial time [7] (Lemma 10) and we showed that the same is true for the class of tree-child networks (Lemma 8). Interestingly, the class of tree-child networks is contained in the class of orchard networks [14] which is in turn contained in the class of tree-based networks [12]. The reason for such an NP-hardness sandwich can perhaps be attributed to the lack of forbidden shapes. Leaf additions to obtain tree-child or tree-based networks target certain forbidden shapes in the network. In the case of tree-child networks, we add a leaf to exactly one outgoing arc of each omnian; for tree-based networks, we add a leaf to exactly one arc of each W-fence. The problem of finding a characterization of orchard networks in terms of (local) forbidden shapes has been elusive thus far [14] - perhaps the NP-hardness result for the orchard variant of the problem indicates that finding such a characterization for orchard networks may not be possible.

One can also consider the leaf addition problem for non-binary networks. Non-binary networks generalize the networks considered in this paper by allowing vertices to have varying indegrees and outdegrees. This generalized problem remains NP-complete since the binary version is a specific case. It could be interesting to try to find an MILP formulation for the nonbinary version.

Another natural research direction is to consider different proximity measures. One that may be of particular interest is a proximity measure based on arc deletions. That is, what is the minimum number of reticulate arc deletions needed to make a network orchard? Susanna showed that this measure is incomparable to the leaf addition proximity measure [17], yet it is not known if it is also NP-hard to compute.

References

1 Mihaela Baroni, Charles Semple, and Mike Steel. Hybrids in real time. *Systematic biology*, 55(1):46–56, 2006.

2 Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021. URL: `http://www.optimization-online.org/DB_HTML/2021/12/8728.html`.

3 Gabriel Cardona, Francesc Rosselló, and Gabriel Valiente. Comparison of tree-child phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(4):552–569, 2008.

4 Péter L Erdős, Charles Semple, and Mike Steel. A class of phylogenetic networks reconstructable from ancestral profiles. *Mathematical biosciences*, 313:33–40, 2019.

5 Mareike Fischer and Andrew Francis. How tree-based is my network? Proximity measures for unrooted phylogenetic networks. *Discrete Applied Mathematics*, 283:98–114, 2020.

6 Mareike Fischer, Tom Niklas Hamann, and Kristina Wicke. How far is my network from being edge-based? Proximity measures for edge-basedness of unrooted phylogenetic networks. *arXiv preprint*, 2022. `arXiv:2207.01370`.

7 Andrew Francis, Charles Semple, and Mike Steel. New characterisations of tree-based networks and proximity measures. *Advances in Applied Mathematics*, 93:93–107, 2018.

8 Andrew R Francis and Mike Steel. Which phylogenetic networks are merely trees with additional arcs? *Systematic biology*, 64(5):768–777, 2015.

9 Benjamin E Goulet, Federico Roda, and Robin Hopkins. Hybridization in plants: old ideas, new techniques. *Plant physiology*, 173(1):65–78, 2017.

10 Carlton Gyles and Patrick Boerlin. Horizontally transferred genetic elements and their role in pathogenesis of bacterial disease. *Veterinary pathology*, 51(2):328–340, 2014.

11 Momoko Hayamizu. A structure theorem for rooted binary phylogenetic networks and its implications for tree-based networks. *SIAM Journal on Discrete Mathematics*, 35(4):2490–2516, 2021.

12 Katharina T Huber, Leo van Iersel, Remie Janssen, Mark Jones, Vincent Moulton, Yukihiro Murakami, and Charles Semple. Orienting undirected phylogenetic networks. *arXiv preprint*, 2019. `arXiv:1906.07430`.

13 Remie Janssen and Pengyu Liu. Comparing the topology of phylogenetic network generators. *Journal of Bioinformatics and Computational Biology*, 19(06):2140012, 2021.

14 Remie Janssen and Yukihiro Murakami. On cherry-picking and network containment. *Theoretical Computer Science*, 856:121–150, 2021.

15 Laura Jetten and Leo van Iersel. Nonbinary tree-based phylogenetic networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(1):205–217, 2016.

16 Fabio Pardi and Celine Scornavacca. Reconstructible phylogenetic networks: do not distinguish the indistinguishable. *PLoS computational biology*, 11(4):e1004135, 2015.

17 Merel Susanna. Making phylogenetic networks orchard: Algorithms to determine if a phylogenetic network is orchard and to transform non-orchard to orchard networks. Bachelor's thesis, Delft University of Technology, 2022. `http://resolver.tudelft.nl/uuid:724ac2af-e569-4586-b367-288fef890252`.

18 Gergely J Szöllősi, Adrián Arellano Davín, Eric Tannier, Vincent Daubin, and Bastien Boussau. Genome-scale phylogenetic analysis finds extensive gene transfer among fungi. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 370(1678):20140335, 2015.

19 Leo van Iersel, Remie Janssen, Mark Jones, and Yukihiro Murakami. Orchard networks are trees with additional horizontal arcs. *Bulletin of Mathematical Biology*, 84(8):1–21, 2022.
20 Leo van Iersel, Remie Janssen, Mark Jones, Yukihiro Murakami, and Norbert Zeh. A unifying characterization of tree-based networks and orchard networks using cherry covers. *Advances in Applied Mathematics*, 129:102222, 2021.
21 David A Wickell and Fay-Wei Li. On the evolutionary significance of horizontal gene transfers in plants. *New Phytologist*, 225(1):113–117, 2020.
22 Chi Zhang, Huw A Ogilvie, Alexei J Drummond, and Tanja Stadler. Bayesian inference of species networks from multilocus sequence data. *Molecular biology and evolution*, 35(2):504–517, 2018.
23 Louxin Zhang. On tree-based phylogenetic networks. *Journal of Computational Biology*, 23(7):553–565, 2016.

A Appendix

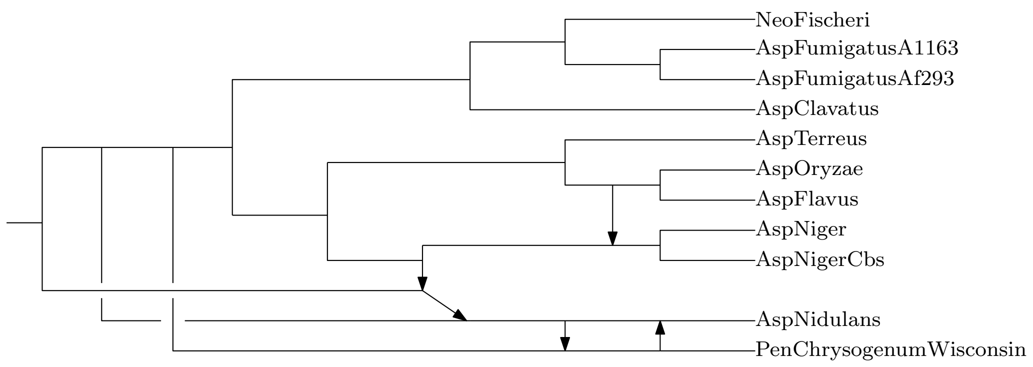


Figure 8 The network of Figure 1 without the added leaf. Observe that there exists no HGT-consistent labelling for the network, by the arguments provided in Remark 22.

▶ Remark 22. We first elaborate on why we need an added leaf (*unsampled taxon*) in the network of Figure 1 to ensure that the network admits an HGT-consistent labelling. We know that a network has an HGT-consistent labelling if and only if it is orchard (Theorem 2). Let N be the network without *unsampled taxon* (see Figure 8). We will show that N is not orchard. To see this, note that the order in which cherries and reticulated cherries are reduced does not matter [14]. This means that if N were orchard, then there would exist a cherry picking sequence starting with

$$(AspNidulans, PenChrysogenumWisconsin)(PenChrysogenumWisconsin, AspNidulans).$$

After reducing these cherries, the distance between the leaf $AspNidulans$ and any other leaf remains of distance at least 4, regardless of other reductions that take place in the network. This shows that the network cannot be orchard, and therefore the network cannot have an HGT-consistent labelling.

▶ **Lemma 7.** *Let N be a network. Then $L_{\mathcal{TC}}(N)$ is equal to the number of omnians. Moreover, N can be made tree-child by adding a leaf to exactly one outgoing arc of each omnian.*

Proof. By definition, a network is tree-child if and only if it contains no omnians. We show that every leaf addition can result in a network with one omnian fewer than that of the original network. Let uv be an arc where u is an omnian. Add a leaf x to uv. In the resulting network, u has a child (the parent of x) that is a tree vertex, and it is no longer an omnian. The newly added tree vertex has a leaf child x; the parent-child combinations remain unchanged for the rest of the network, so at most one omnian (in this case u) can be removed per leaf addition. It follows that $L_{\mathcal{TC}}(N)$ is at least the number of omnians in N. By targeting arcs with omnian tails, we can remove at least one omnian per every leaf addition, so that $L_{\mathcal{TC}}(N)$ is at most the number of omnians in N. Therefore, $L_{\mathcal{TC}}(N)$ is exactly the number of omnians in N. ◀

▶ **Lemma 8.** *Let N be a network. Then $L_{\mathcal{TC}}(N)$ can be computed in $O(|N|)$ time.*

Proof. We first show that the number of omnians of N can be computed in $O(|N|)$ time, by checking, for each vertex, the indegrees of its children. A vertex is an omnian if and only if all of its children are of indegree-2. Since the degree of every vertex is at most 3, each search within the for loop takes constant time. The for loop iterates over the vertex set which is of size $O(N)$. By Lemma 7, since $L_{\mathcal{TC}}(N)$ is the number of omnians in N, we can compute $L_{\mathcal{TC}}(N)$ in $O(|N|)$ time. ◀

It has been shown already that $L_{\mathcal{TB}}(N)$ can be computed in $O(|N|^{3/2})$ time where $|N|$ is the number of vertices in N [7]. We show that this can in fact be computed in $O(|N|)$ time.

▶ **Lemma 9.** *Let N be a network. Then $L_{\mathcal{TB}}(N)$ is equal to the number of W-fences. Moreover, N can be made tree-based by adding a leaf to any arc in each W-fence in N.*

Proof. By Lemma 4, a network is tree-based if and only if it contains no W-fences. We show that every leaf addition can result in a network with one W-fence fewer than that of the original network. Suppose that N contains at least one W-fence. Otherwise we may conclude that the network is tree-based by Lemma 4. Let $(a_1, a_2, \dots, a_k)$ be a W-fence in N where $a_i = u_iv_i$ for $i \in [k]$, and add a leaf x to a_1; let p_x be the tree vertex parent of x. In the resulting network, the arcs in $\{u_1p_x, p_xv_1, p_xx, a_2, a_3, a_4, \dots, a_k\}$ are decomposed into their unique maximal zig-zag trails (Theorem 5) as two N-fences (u_1p_x) and $(a_k, a_{k-1}, \dots, a_3, a_2, p_xv_1, p_xx)$. All other arcs remain in the same maximal zig-zag trails as that of N. Therefore the number of W-fences has gone down by exactly one. This can be repeated for all W-fences in the network; it follows that $L_{\mathcal{TB}}(N)$ is the number of W-fences in N.

A quick check shows that adding a leaf to any arc in the W-fence decomposes the W-fence into two N-fences. ◀

▶ **Lemma 10.** *Let N be a network. Then $L_{\mathcal{TB}}(N)$ can be computed in $O(|N|)$ time.*

Proof. Finding the maximal zig-zag decomposition takes $O(|N|)$ time (Proposition 5.1 of [11]). Counting the number of W-fences in the decomposition gives $L_{\mathcal{TB}}(N)$ by Lemma 9. ◀

▶ **Observation 12.** *Let G be a 3-regular graph and let N_G be the network obtained by the reduction. Then N_G is tree-based.*

Proof. It is easy to check that the arcs of N_G are decomposed into M-fences and N-fences (the principal part of each gadget Gad(v) is an N-fence; each arc leaving the principal part of a gadget Gad(v) is an N-fence of length 1; the remaining arcs decompose into M-fences of length 2). By Lemma 4, N_G must be tree-based. ◀

▶ **Lemma 13.** *Let N be a tree-based network, and let $\hat{N}_1, \hat{N}_2, \dots, \hat{N}_n$ denote the N-fences of N of length at least 3. Then every cherry cover of N contains the reticulated cherry shapes $\{(\mathrm{head}(a^i_{2j-1})c^i_{2j-1}), a^i_{2j}, a^i_{2j+1}\}$ for $i \in [n]$ and $j \in \left[\frac{k_i-1}{2}\right]$.*

Proof. Let $\hat{N}_i = (a^i_1, a^i_2, \dots, a^i_{k_i})$ be an N-fence of length $k_i \geq 3$. Observe that in every cherry cover, exactly one incoming arc of every reticulation is covered by a reticulated cherry shape as a middle arc (since the network is binary; for non-binary networks, this is not true in general [20]). Since $\text{head}(a^i_1)$ is a reticulation, one of a^i_1 or a^i_2 must be in a reticulated cherry shape as a middle arc. But $\text{tail}(a^i_1)$ is a reticulation; therefore, a^i_2 must be in a middle arc of a reticulated cherry shape. The other two arcs of the same reticulated cherry shapes are then fixed to be $\text{head}(a^i_1)c^i_1$ and a^i_3. Repeating this argument for the reticulations $\text{head}(a^i_{2j+1})$ for $j \in \left[\frac{k_i-1}{2}\right]$ gives the required claim for the N-fence $\hat{N}_i$; further repeating this argument for every N-fence gives the required claim. ◀

▶ **Lemma 14.** *Let N be a tree-based network and suppose that for two N-fences $\hat{N}_u := (a^u_1, a^u_2, \dots, a^u_{k_u})$ and $\hat{N}_v := (a^v_1, a^v_2, \dots, a^v_{k_v})$ of length at least 3, there exist directed paths in N from $\text{head}(a^u_h)$ to $\text{tail}(a^v_i)$ and from $\text{head}(a^v_j)$ to $\text{tail}(a^u_k)$, for even h, i, j, k with $k < h$ and $i < j$. Then every cherry cover auxiliary graph of N contains a cycle.*

Proof. Let us again denote the reticulated cherry shape that contains a^u_h and a^u_{h+1} by $R^u_{h/2}$, and similarly for $R^v_{i/2}$, $R^v_{j/2}$,and $R^u_{k/2}$. By Lemma 13, all of $R^u_{h/2}$, $R^v_{i/2}$, $R^v_{j/2}$, $R^u_{k/2}$ appear in the cherry cover auxiliary graph. Moreover $R^u_{k/2}$ is above $R^u_{h/2}$, and $R^v_{i/2}$ is above $R^v_{j/2}$. Now observe that for any consecutive arcs on the path from $\text{head}(a^u_h)$ to $\text{tail}(a^v_i)$, either they are part of the same reticulated cherry shape in the cherry cover, or they are part of different cherry shapes with one cherry shape directly above the other. This implies that there is a path from $R^u_{h/2}$ to $R^v_{i/2}$ in the cherry cover auxiliary graph. A similar argument shows that there is a path from $R^v_{j/2}$ to $R^u_{k/2}$. But then we have that $R^u_{h/2}$ is above $R^v_{i/2}$, which is above $R^v_{j/2}$, which is above $R^u_{k/2}$, which is above $R^u_{h/2}$ and we have a cycle. ◀

▶ **Lemma 15.** *Let G be a 3-regular graph and let N_G be the network obtained by the reduction above. Suppose that A is a set of arcs of N_G, for which adding leaves to every arc in A results in an orchard network. For every edge $uv \in E(G)$, there exists an arc $a \in A$ that is an arc of the principal part of* $\text{Gad}(u)$ *or* $\text{Gad}(v)$*.*

Proof. We prove this lemma by contraposition. Let us assume that there is an edge $uv \in E(G)$, such that no arcs of the principal part of $\text{Gad}(u)$ or $\text{Gad}(v)$ are in A. We shall show that the network obtained by adding leaves to all $a \in A$ in N_G – which we denote $N_G + A$ – is not orchard.

From Theorem 1 we know that N_G is orchard if and only if N_G has an acyclic cherry cover. We show here that $N_G + A$ will not have an acyclic cherry cover, thereby showing that $N_G + A$ is not orchard.

As no arcs were added to the principal part of $\text{Gad}(u)$ or $\text{Gad}(v)$, these principal parts remain N-fences in $N_G + A$. Furthermore by construction N has an arc from some $\text{head}(a^v_h)$ to $\text{tail}(a^u_i)$ for even $h \geq 10$ and even $i \leq 6$, and so $N_G + A$ has a path from $\text{head}(a^v_h)$ to $\text{tail}(a^u_i)$. Similarly $N_G + A$ has a path from $\text{head}(a^u_j)$ to $\text{tail}(a^v_k)$ for some even $j \geq 10$ and $h \leq 6$. Then Lemma 14 implies that the auxiliary graph of any cherry cover of $N_G + A$ contains a cycle. By Theorem 1, we have that $N_G + A$ is not orchard. ◀

▶ **Lemma 16.** *Let G be a 3-regular graph and let N_G be the network obtained by the reduction described above. Then G has a minimum vertex cover of size k if and only if $L_{\mathcal{OR}}(N_G) = k$.*

Proof. Suppose first that V_{sol} is a vertex cover of G with at most k vertices. We shall show that adding a leaf to an arc of the principal part of each $\text{Gad}(v)$ for $v \in V_{sol}$ makes N_G orchard. This will show that the minimum vertex cover of G is at least $L_{\mathcal{OR}}(N_G)$. In the remainder of this proof, we will refer to vertices and arcs of N_G as introduced above in the reduction.

For every $v \in V_{sol}$, we add a leaf z^v to the arc $w^v_4 r^v_4$ of Gad(v) (see Figure 5c). Let q^v be the parent of z^v. The key idea is that this splits the principal part of Gad(v) from an N-fence into an N-fence and an M-fence, and this allows us to avoid the cycle in the cherry cover auxiliary graph (see Figure 5d).

Let us call the new network M. To formally show that M is orchard, we give an HGT-consistent labelling $t : V(M) \to \mathbb{R}$.

Begin by setting $t(\rho) = 0$, and for any vertex in $s_1, \ldots, s_{g-1}$ or $\rho^v, m^v_4, \ldots, m^v_2$ for any v in $V(G)$, let this vertex have label equal to the label of its parent plus 1. Let h be the maximum value assigned to a vertex so far, and now adjust t by subtracting $(h + 1)$ from each label. Thus, we may now assume that all vertices in $\rho, s_1, \ldots, s_{g-1}$ or $m^v_4, \ldots, m^v_2$ for any v in $V(G)$ have label ≤ -1. Now set $t(m^v_1) = 0$ and $t(r^v_0) = 0$, for each v in $V(G)$.

It is easy to see that so far t satisfies the properties of an HGT-consistent labelling. It remains to label the vertices in the principal part of each gadget Gad(v), and the leaves of each gadget, and the new vertices q^v and z^v for $v \in V_{sol}$. We do this as follows.

For $v \in V_{sol}$, set $t(r^v_1) = t(w^v_1) = 12, t(r^v_2) = t(w^v_2) = 13, t(r^v_3) = t(w^v_3) = 14$, and $t(r^v_4) = t(q^v) = 15$. Set $t(w^v_4) = 1, t(r^v_5) = t(w^v_5) = 2, t(r^v_6) = t(w^v_6) = 3$, and $t(r^v_7) = t(w^v_7) = 4$.

For $v \notin V_{sol}$, set $t(r^v_1) = t(w^v_1) = 5$, and $t(r^v_i) = t(w^v_i) = i + 4$ for every i up to $t(r^v_7) = t(w^i_7) = 11$.

Finally, for each leaf ℓ with parent p set $t(\ell) = t(p) + 1$.

It remains to observe that t is a non-temporal labelling of M and for every reticulation r in M, r has exactly one parent p with $t(p) = t(r)$. Thus t is an HGT-consistent labelling of M, and it follows from Theorem 2 that M is orchard.

Suppose now that we have a set of arcs A_{sol} of N_G of size at most k, such that adding leaves to the arcs in A_{sol} makes N_G orchard. By Lemma 15, for every edge $uv \in E(G)$, there exists an arc $a \in A_{sol}$ that is an arc of the principal part of Gad(u) or Gad(v). It follows immediately that the set $\{v \in V(G) : A_{sol}$ contains an arc of the principal part of Gad$(v)\}$ is a vertex cover of G. Since this is true for any such set of arcs A_{sol}, it follows that if there is such an A_{sol} of size at most k, then there must exist a vertex cover of G of size at most k. ◀